PandaSearch: a Fine-grained Academic Search Engine for Research Documents

Feiran Huang [1], Jia Li [1], Jiaheng Lu [1], Tok Wang Ling [2], Zhaoan Dong [1]
[1] *DEKE and School of Information, Renmin University of China*
[2] *School of Computing, National University of Singapore*

***Abstract*—In the world of academia, research documents enable the sharing and dissemination of scientific discoveries. During these "big data" times, academic search engines are widely used to find the relevant research documents. Considering the domain of computer science, a researcher often inputs a query with a specific goal to find an algorithm or a theorem. However, to this date, the return result of most search engines is just as a list of related papers. Users have to browse the results, download the interesting papers and look for the desired information, which is obviously laborious and inefficient. In this paper, we present a novel academic search system, called `PandaSearch`, that returns the results with a fine-grained interface, where the results are well organized by different categories, such as definitions, theorems, lemmas, algorithms and figures. The key technical challenges in our system include the automatic identification and extraction of different parts in a research document, the discovery of the main topic phrases for a definition or a theorem, and the recommendation of related definitions or figures to elegantly satisfy the search intention of users. Based on this, we have built a user friendly search interface for users to conveniently explore the documents, and find the relevant information.**

I. INTRODUCTION

In the world of academia, search engines are widely used to find the relevant research documents. There are many tools and systems which help to find relevant papers, such as DBLP, PubMed, arXiv, Citeseer, Google Scholar, ACM Digital Library and IEEE CS Digital library. These present a search engine like interface which allows users to search papers via keywords, or via faceted search on features such as author, venue etc. Although this approach provides good search results in many cases, it is far from optimal since it does not directly provide the exact information users want. For example, a user submits "DFS algorithm" to search a Depth-First-Search algorithm on graphs, or "Differential privacy" for definitions about differential privacy. In the existing search engine, the returned result is just a list of related papers. Users have to browse the results, download the interested papers and search the desired information, which is obviously laborious and inefficient. In this paper, our goal is to build a novel search engine for various fine-grained information, such as definitions, algorithms, lemmas, figures et al. , which are called "*knowledge cells*" hereafter, to enable a researcher to conveniently find the information.

To support this aim, we have built a fine-grained search engine, to aid the specific type of information discovery and analysis. There are some challenges to overcome in meeting this goal. The first one is to correctly identify the boundary of each *knowledge cell*. The structures of papers can vary greatly, and the correct identification of the region of each knowledge cell is a challenging job. In our system, we build a boundary identifier to automatically identify the region of a knowledge cell.

DEFINITION 2.1 (DIVERGENT DESIGN). *Given a workload $W = Q \cup U$, a weight function f, and a number of replicas n, a divergent design corresponds to a set $p = \{(W_1, f_1), \ldots, (W_n, f_n)\}$, such that:*

- $W_i = Q_i \cup U$ *for all* $i \in [1, n]$, *where* $Q_1 \cup \cdots \cup Q_n = Q$.
- $f_i(u) = f(u)$ *for all* $u \in U$.
- $\sum_{i \in [1,n]} f_i(q) = f(q)$.

The configuration of each replica is computed as $I_i = DBAdv(W_i, f_i, b)$.

Fig. 1. Example of an extracted definition

The second challenge is to discover the topics (or key phrases) of each knowledge cell to facilitate users in searching information. Some definitions or theorems highlight their topics at the beginning. For example, the topic of the definition in Figure 1 is clearly shown in the parenthesis at the beginning of the definition (i.e. "DIVERGENT DESIGN"), but in Figure 2, the topic is hidden in the textual description. The topic of a knowledge cell may appear in different places, such as the caption of a figure, the specification within a table, and even reference context which refers to an algorithm in the main body of the document; this makes it difficult to apply a simple solution for topic extraction.

The third challenge is to display knowledge cells according to users' queries. The scientific community has converged on the Portable Document Format (PDF) as the de facto standard for sharing research papers. But the current PDF analytic tools (such as PDFBox) do not provide interfaces to extract a part of one pdf page. Therefore, it is a challenge to correctly extract and display knowledge cells from PDF files.

The fourth challenge is to effectively rank the results. The existing system ranks the results (i.e. papers) according to publishing years (e.g. DBLP, Microsoft academic search) or citation numbers (e.g. Google Scholar). But both of these criterions are not suitable to our application, since a document with more citations does not necessarily mean that all of its knowledge cells are more important. Due to the above reasons, the current ranking strategy is not applicable for our case.

In this demonstration, we propose a framework to effectively identify different categories of knowledge cells in research

Definition 3 We define the freshness of element e_i *averaged over time*, $\bar{F}(e_i)$, and the freshness of database S *averaged over time*, $\bar{F}(S)$, as

$$\bar{F}(e_i) = \lim_{t\to\infty} \frac{1}{t}\int_0^t F(e_i;t)dt$$

$$\bar{F}(S) = \lim_{t\to\infty} \frac{1}{t}\int_0^t F(S;t)dt.$$

The time average of age can be defined similarly. □

Fig. 2. Another example of a definition, whose topic is hidden in the description

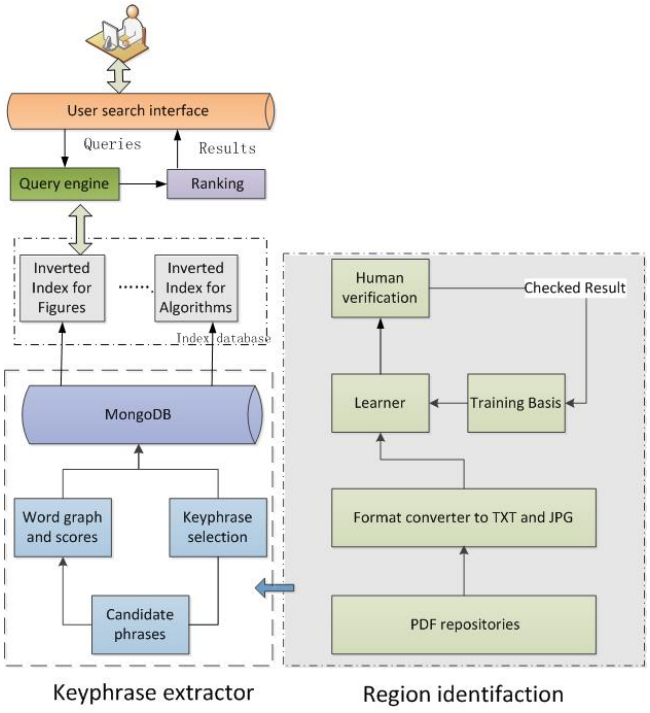

Fig. 3. System Framework

documents for the domain of computer science, and provide a user friendly interface for searching and browsing them to address the above challenges. Our system can be accessed online at: www.cspandasearch.net.

II. SYSTEM OVERVIEW

Figure 3 shows the framework of our system. It consists of three main components. In the *region identification* stage, papers are collected and analyzed. The different knowledge cells are identified and extracted. The next stage is about *keyphrase extraction*. The keyphrase of each knowledge cell is extracted, which is used for keyword search later. Finally, the *search and recommender engine* (including user interface, query engine, ranking and index databases) allows users to search knowledge cells and explore the related information. Below, we describe these three steps in details.

Paper collection and converting We collected a large corpus of documents by crawling public websites. As a preprocessing step, from each paper we converted all papers from PDF format

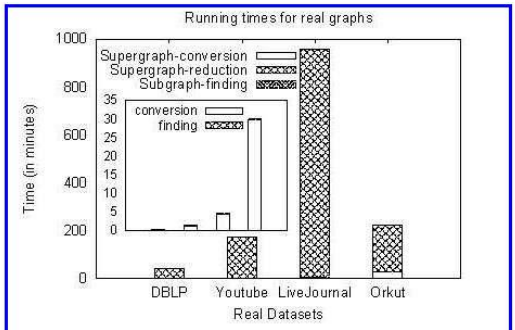

Figure 2: Running time for large real graphs.

Dataset	Nodes	Edges	Avg. Degree
com-DBLP	317,080	1,049,866	3.31
com-Youtube	1,134,890	2,987,624	2.63
com-LiveJournal	3,997,962	34,681,189	8.67
com-Orkut	3,072,441	117,185,083	38.14

Table 7: Large real graphs.

Fig. 4. Example for the mixture of figures and tables

to both text and image JPG files for processing, because text files can be used for keyword search, but they cannot keep the original appearance of figures and formulas. On the other hand, the image files can keep the original structure and shape, but cannot be used for keyword match. Therefore, we make use of two different formats for the purpose of search and display respectively. In total, we obtained a collection of 12,946 papers mainly from the computer science domain.

A. Region Identification of Knowledge Cells

Given the wide variety of styles for the region of a knowledge cell, we need to develop an automatic algorithm to identify and extract them for the purpose of display and analysis. Some knowledge cells such as definitions, theorems or lemmas often start with the bold font at the beginning of a paragraph. See the example in Figure 1 or 2, where a definition is started with the keyword "**Definition**"; this is easy to recognize. But in the example of Figure 4, a figure is displayed together with a table such that it is not easy for machines to distinguish between a figure and a table. As a result, we built a boundary detector with active learning to automatically extract a knowledge cell.

Training data sets We need a training set to accurately train our detector to find the region of a knowledge cell. However, the process of manually labeling thousands of documents is a very time-consuming one. Therefore, we use active learning [1] to speed up this process to generate a high quality training set much more efficiently. In our implementation, we built a training set of 1000 documents, containing 13,876 knowledge cells which are labeled.

Boundary detection Our detector uses a large number of features to decide the start and end points of a knowledge cell. The main features are derived from the context around each boundary in question. For example, the features to indicate a "*Definition*" cell include (a) whether a paragraph starts with

a word "*Definition*", (b) whether the first letter is capitalized, and (c) whether the word is followed by a number. Similarly, for each type of a knowledge cell, we can generate a number of features for machine learning algorithms. We make use of the feature based, open source libSVM classifier [2] to identify from the possible regions of knowledge cells in documents. We evaluated the precision of this detector in the way that we score this as correct if the detector correctly finds the boundary and extracts the knowledge cell. Under this measure, for example, our classifier achieves 78% precision, 72% recall for definitions, and 84% precision, 75% recall for algorithms, which are sufficient for the purpose of this demonstration.

B. Keyphrases Extraction from Knowledge Cells

We now discuss our approach to the identification of keyphrases for different knowledge cells. We built a topic generator by extending the ideas of CiteTextRank [3], which is a keyphrase extraction algorithm. For our purpose, we apply some changes to this algorithm. Since we extract keyphrases for different knowledge cells in one document, we do not use the citation among documents, whereas we use the reference context of knowledge cells within one document. For example, one reference context in a research paper looks like "*Figure 4 also shows MRR plots comparing CTR models*", which indicates that Figure 4 talks about "*MRR*" and "*CTR models*".

Given a target knowledge cell K, the objective of the keyphrase extraction task is to extract a ranked list of candidate words or phrases from K that best represent K. Algorithms ([3]) for unsupervised keyphrase extraction involve the three steps. Before we describe the algorithm in details, we first give a definition about the context of a knowledge cell.

Definition 1: (**Context of a knowledge cell**) Given a document D and a knowledge cell K in D, a *referred context* of K is defined as a context in which K is referred in D. In addition, there are two types of context regarding to the own description of K, one is called the *key context*, which summarizes the topic of K, such as the title of an algorithm, or the caption of a figure; Otherwise it is a *plain context*.

Candidate words extraction Candidate words or lexical units are extracted from the context of the target knowledge cells (see Definition 1) by applying stopword and parts-of-speech filters. If a knowledge cell K is referred in multiple contexts, then we aggregate all such contexts and simply refer to them as the referred context (as applicable). Only nouns and adjectives that are likely to be keyphrases are retained in this step.

Candidate words score Next, candidate words are scored based on the following criterion. Let T represent the types of available contexts of K. We construct an undirected graph G = (V;E) for K as follows:

1. For each unique candidate word extracted from all available contexts of K, add a vertex in G;

2. Add an undirected edge between two vertices v_i and v_j if the words corresponding to these vertices occur within a window of n continuous tokens in any of the contexts.

3. The weight w_{ij} of an edge (v_i,v_j) $\in$ E is given as

$$w_{ij} = w_{ji} = \sum_{t \in T} \sum_{c \in C_t} \gamma_t \cdot sim(c,k) \cdot Occur_c(v_i, v_j) \quad (1)$$

where $sim(c,k)$ is the cosine similarity between the tf-idf vectors of any context c of k and k; Occur$_c$(v_i,v_j) is the co-occurrence frequency of words corresponding to v_i and v_j in context c, C_t is the set of contexts of type $t \in T$; and γ_t is the weight for contexts of type t. We incorporate the notion of "*importance*" of contexts of a certain type using the γ_t parameters. For instance, one might assign higher importance to key context and reference context over plain context.

4. We score the vertices in G (and the corresponding candidate words) using the PageRank algorithm.

Keyphrases selection Finally, consecutive words, phrases or n-grams are scored by using the sum of scores of individual words that comprise the phrase [4]. The top-scoring phrases are output as predictions (the keyphrases for the document).

C. Ranking of knowledge cells

In this subsection, we discuss how to extend the model of Random Walk with Restart (for short RWR, also known as personalized PageRank) [5] to assign a weight for each knowledge cell to facilitate the ranking of the search results. As mentioned in Introduction, the existing algorithms use the publishing years and/or the citation numbers to rank the results, which may not be suitable here, because both criterions do not reflect the importance of a particular knowledge cell.

RWR can be briefly described as follows. From a node q, the random walker can walk to its neighbors with probabilities proportional to the edge weights. In each step, it has a probability of c to return to q, where c is a constant. RWR can be defined recursively as

$$w_q = (1-c) \sum_{j \in Adj(q)} p_{j,q} \cdot w_j + c \quad (2)$$

where c (0$<$ c $<$1) is the constant restart probability, and

$$p_{j,q} = sim(j, A) \cdot NormCiteNum(j, q). \quad (3)$$

In our application, each node represents a knowledge cell. $j \in Adj(q)$ means that the paper of j cites that of q. Further, $NormCiteNum(j,q)$ denotes the normalized number of citations of q appeared in j, which reflects the importance of the paper of q with respect to that of j. Note that $sim(j,A)$ denotes the cosine similarity between the tf-idf vectors of the keyphrases of j and that of the abstract of the paper which the cell j belongs to. The intuition in Equation (3) is that (i) a knowledge cell should be assigned a higher weight if it has more overlapping words with the abstract, which illustrates the main idea of the paper; and (ii) a knowledge cell whose paper has more citations should be assigned a higher weight.

III. SYSTEM DEMONSTRATION

Our demo will focus on two scenarios with respect to the discovery and exploration of multiple types of knowledge cells.

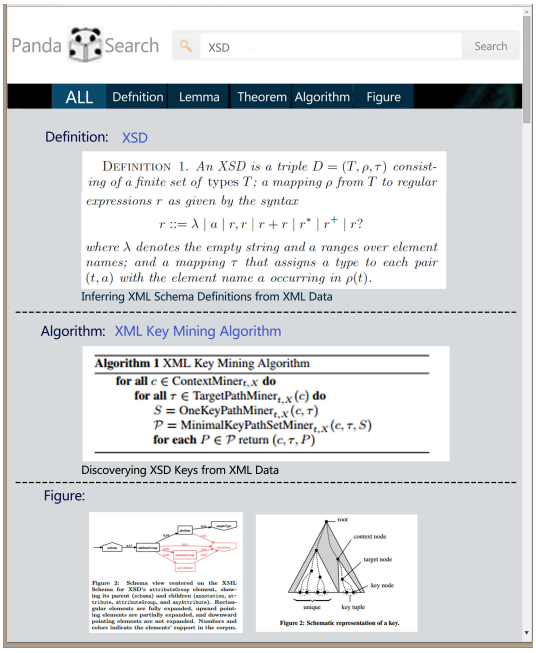

Fig. 5. The result interface to process a query "XSD".

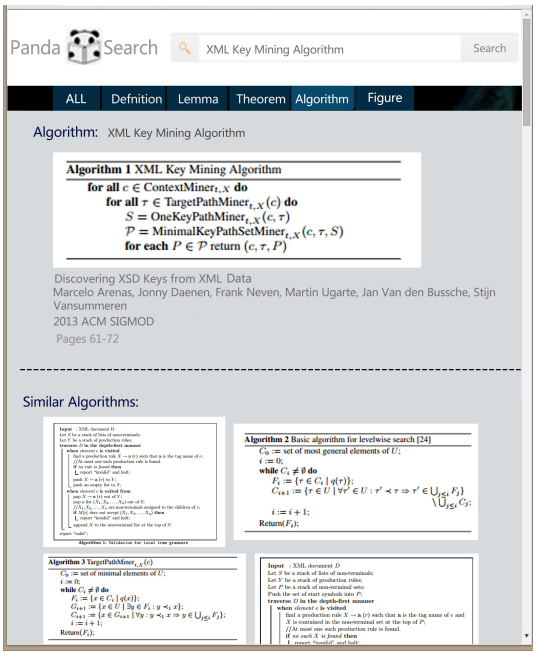

Fig. 6. Browsing algorithms about "XML Key Mining Algorithm".

A. Information discovery

The first function we provide allows searching for a knowledge cell in the domain of computer science, such as "XSD" shown in Figure 5, which means "XML Schema Definitions". Users can input keywords to retrieve a list of definitions, lemmas, theorems, algorithms, and figures to obtain exactly the information they want to analyze. For ease of exploration, each returned result is shown with snippets indicating the paper title, which users can click to see the detailed information about this paper.

B. Information exploration

Then we allow further exploration to see more related knowledge cells. For example, when users are browsing the "XML key mining" algorithm in Figure 5, they can focus on "XML key mining" by clicking the algorithm button to see more related algorithms which appear in other papers. This switches to the search by the algorithm interface, as shown in Figure 6. Likewise, when a user is looking through the interface with respect to some specific pictures in Figure 5, they can easily locate the other figures used in same or the other related documents by clicking the figure button. In this way, users are able to explore the different knowledge cells broadly and deeply by following the found knowledge cell links provided by the system. We will invite our audience to try out our system, search different knowledge cells, and make discoveries for new theorems, figures, definitions, etc.

IV. Related Systems and Conclusion

A number of tools (e.g. [6]) have been built to assist researchers and students to effectively search for relevant literature. Google Scholar, Microsoft Academic Search, Arnet-Miner and CiteSeerX are some of the existing websites which provide detailed information of a paper. Users interact with these websites by manually submitting queries and evaluating the produced output. While all existing sites provide useful and relevant functionality, the motivation and contributions of our system are different: our tool provides an end-to-end solution which, given a keyword search, identifies the relevant knowledge cells, including definitions, lemmas, figures et al., to make them available to users. Our demonstration represents a step forward in practice for handling a general problem about content analysis of research documents. Our system can be accessed online at: www.cspandasearch.net.

Acknowledgement

This paper is partially supported by 863 National High-tech Research Plan of China (No. 2012AA011001), RUC Research Funds (no. 11XNJ003) and NSSF, China (No: 12&ZD220). We are very grateful to Qi Cao and Haiyang Zhao for their help in data processing and system developing.

References

[1] E. Lughofer, "Single-pass active learning with conflict and ignorance," *Evolving Systems*, vol. 3, no. 4, pp. 251–271, 2012.

[2] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM TIST*, vol. 2, no. 3, p. 27, 2011.

[3] S. D. Gollapalli and C. Caragea, "Extracting keyphrases from research papers using citation networks," in *AAAI*, 2014, pp. 1629–1635.

[4] X. Wan and J. Xiao, "Single document keyphrase extraction using neighborhood knowledge." in *AAAI*, 2008, pp. 855–860.

[5] Y. Wu, R. Jin, and X. Zhang, "Fast and unified local search for random walk based k-nearest-neighbor query in large graphs," in *SIGMOD Conference*, 2014, pp. 1139–1150.

[6] M. Lu, S. Bangalore, G. Cormode, M. Hadjieleftheriou, and D. Srivastava, "A dataset search engine for the research document corpus," in *ICDE*, 2012, pp. 1237–1240.